

Revisiting Deep Image Smoothing and Intrinsic Image Decomposition

Qingnan Fan¹ David Wipf² Gang Hua² Baoquan Chen¹

¹Shandong University ²Microsoft Research Asia

fqnchina@gmail.com, {davidwip, ganghua}@microsoft.com, baoquan@sdu.edu.cn

Abstract

We propose an image smoothing approximation and intrinsic image decomposition method based on a modified convolutional neural network architecture with large receptive fields applied directly to the original color image. When training a deep model for these purposes however, it is quite difficult to generate edge-preserving images without undesirable color differences or boundary artifacts. To overcome these obstacles, we supervise intermediate outputs from different functional components during training. For example, we apply both image gradient supervision and a channel-wise rescaling layer that computes a minimum mean-squared error color correction. Additionally, to enhance piece-wise constant effects for image smoothing, we append a domain transform filter with a predicted refined edge map. The resulting deep model, which can be trained end-to-end, directly learns edge-preserving smooth images and intrinsic image decompositions without any special design, specialized auxiliary training data, or input scaling/size requirements. Moreover, our method shows much better numerical and visual results on both tasks and runs in comparable test time to existing deep methods.

1. Introduction

Image smoothing is a fundamental building block of many computer vision and graphics applications including texture removal, image enhancement, edge extraction, image abstraction, and beyond. The underlying goal of image smoothing, or edge preserving filtering, is to extract sparse salient structures like perceptually important edges and contours, while minimizing color differences in some input image of interest. A vast array of practical filtering algorithms have been proposed for this task using traditional image processing principles and energy functions that operate across both local and global image regions [5, 12, 13, 18, 26, 27, 32, 34, 35, 36]. However, often an expensive, impractical optimization problem must be solved per instance to produce the desired edge-aware filtered images [5, 18].

Recently, deep learning has demonstrated a remarkable ability to replace expensive optimization procedures with cheap computational modules that can be trained offline when given access to representative data, and later deployed at a fraction of the computational budget [15]. For example, given a sufficient training corpus obtained by passing input images through some preferred smoothing filter, we can attempt to learn a deep model that efficiently mimics the filter outputs [33]. Of course considerable care must be taken in designing such a network to reduce training data requirements and to ensure that the trained model is indeed capable of fast, robust inference on new images, and naive, black-box approaches are destined to fail. Perhaps even more importantly, a well-designed network for image filtering can also serve as the foundation for more complex, higher-level vision tasks, without the need for cumbersome integration with an iterative, energy-based filtering approach.

In this regard, one notable application of edge-preserving image smoothing is the computation of intrinsic image decompositions [5], which are useful for material recognition, image-based resurfacing, and relighting, among other things. The intrinsic image model assumes an ideal diffuse environment, in which an input image I is the pixel-wise product of an albedo image A and a shading image S , i.e.,

$$I \approx A \odot S. \quad (1)$$

The albedo or reflectance image indicates how object surface materials reflect light, while shading accounts for illumination effects due to geometry, shadows, and interreflections. Obviously, estimating such a decomposition is a fundamentally ill-posed problem as there exist infinitely many feasible decompositions. Fortunately though, prior information, often instantiated via specially tailored image smoothing filters or energy terms, allows us to constrain the space of feasible solutions [2, 4, 6, 9, 28, 37]. For example, the albedo image will usually be nearly piece-wise constant, with a finite number of levels reflecting a discrete set of materials and boundaries common to natural scenes. In contrast, the shading image is often assumed to be grayscale, and is more likely to contain smooth gradations quantified by small gradients except at locations with cast shadows or

abrupt changes in scene geometry [22].

Naturally, given access to ground truth intrinsic image decompositions, deep networks provide a data-driven candidate for solving this ill-posed inverse problem with fewer heuristic or hand-crafted assumptions. In this paper, we propose a flexible deep neural network architecture that is simultaneously capable of handling wide-ranging, generic image smoothing tasks, where the primary goal is an efficient implementation of existing edge-aware filtering effects, as well as intrinsic image decompositions, where now an augmented objective supports improved accuracy on a higher-level vision task. Our fully trainable, end-to-end network relies on three important ingredients:

Large Flexible Receptive Fields: Edge-aware filtering requires contextual information across a relatively wide image region, especially in the context of an end-to-end system. Additionally, accurate intrinsic image decompositions must account for the fact that distant pixels with the same reflectance on a large surface could have significant color differences due to the accumulation of smoothly-varying soft cast shadows [5]. Hence large receptive fields are essential for both tasks, and we choose to actualize them with maximal flexibility via a fully convolutional subnetwork with many layers.

Color Correction and Gradient Supervision: For image smoothing, the input and output color images should be highly correlated, but unfortunately the color may be attenuated during many forward convolutions in a deep network such as ours. To address this problem in the context of superresolution, [19] adds a skip connection directly from input to output images, which is not effective for image smoothing in our experience. Instead, we include a color correction layer which scales each channel of the image to minimize MSE color differences and naturally allows gradient backpropagation for training purposes. We also observe that supervising gradients of predicted images using an auxiliary objective helps preserve salient edges; however, this is notably different than direct operation in the gradient domain akin to most existing methods.

Domain Transform: To further enhance piece-wise constant effects favored by sparsity-based filtering methods [32, 5], we also leverage a domain transform [14, 8]. Practically speaking, in our system this can be viewed as a series of final, parameterless network layers that implicitly filter color information across the image in a warped domain. This warping is determined by edge maps derived from a preliminary smoothed image, which emerges from the lower convolutional and color correction layers described previously. Although the domain transform layers themselves do not actually increase the overall network capacity per se, they nonetheless alter the space of candidate filters and influence which image regions should be smoothed or not.

Overall, we propose an end-to-end system that can effi-

ciently approximate a variety of computationally intensive image smoothing filters as well as infer related intrinsic image decompositions. For each of these tasks, our method demonstrates state-of-the-art performance both visually and numerically. We differentiate our design choices from existing methods in the next section.

2. Relationship with Existing Models

Image Smoothing: Recently, there have been multiple attempts [33, 25] to imitate the behavior of a large, representative family of image smoothing filters using a single, data-driven approximation framework capable of accelerated inference speeds. For example, [33] proposed a 3-layer network for first predicting prominent gradients, followed by a separate post-processing step involving filter-specific tuning parameters to reconstruct the smoothed image estimate. A potential limitation however, is that gradient or salient structure prediction errors can easily propagate to surrounding areas in the pixel domain during the reconstruction step, reducing visual quality. Moreover, practical image smoothing filters favor varying degrees of edge preservation, spanning the gamut from blurry to piece-wise constant images. Purely gradient-based processing tends to struggle more generating images on the blurry end of this scale, where continuous color transitions may be required. For these reasons, our approach operates entirely in the original color image domain, a gradient supervision term notwithstanding.

A second approach from [25] involves a hybrid network composed of several convolutional layers followed by a set of recurrent network layers. In brief, the CNN portion of the model takes the original image as input and computes a set of weights that are presumably reflective of salient image structures. The recurrent network layers then take these weights, as well as the original image, and act as a form of data-driven guided filter to produce the desired smoothing effect. To a certain degree, our approach can be viewed as the antithesis of this model. Whereas we explicitly parameterize an image model and only later refine it using gradient supervision and a parameterless domain transform modulated via a small subnetwork, the method from [25] fully parameterizes the weights of a recurrent guided filter array, which acts much like a domain transform via the analysis from [8], but then applies these filters only to multi-scale versions of the original raw images. A downside of the later strategy is that it requires downsampling the original images to multiple scales, and at least in its present form, will not work with image sizes that are not multiples of 16 (provided code actually only works with special image sizes). Additionally, any mismatch between the computed weights and legitimate salient edge structure will necessarily propagate to wider regions in the predicted output image since the recurrent filters have no mechanism to compensate.

Intrinsic Image Decomposition: To move beyond

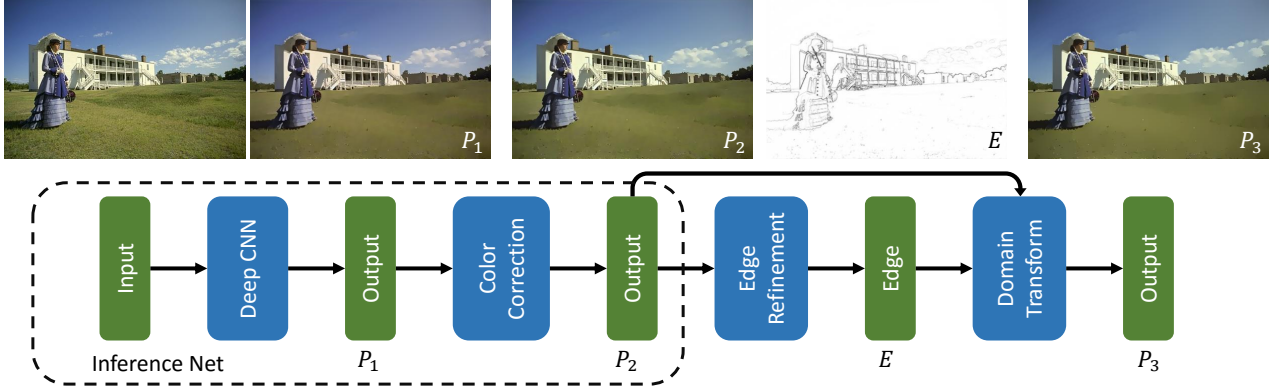


Figure 1. **Basic Network Structure.** For smoothing/filtering tasks, an input image is fed into an inference network (Deep CNN + channel-wise rescaling layer for color correction), followed by a domain transform modulated using an edge refinement step. For intrinsic image decompositions, this pipeline is modified slightly via separate albedo and shading paths as described in the text.

the traditional filtering approaches mentioned in Section 1 to data-driven deep models capable of producing scene-level intrinsic image decompositions, we require images with known ground truth. The MPI-Sintel intrinsic images benchmark [7] provides an important step in this direction, and several existing deep network pipelines have been built upon it. First, [31] learns a two-scale convolutional network to directly predict both albedo and shading images. However, the specific architecture, which closely resembles that from [10] developed for predicting depth and surface normals, involves intermediate feature maps at $1/32$ scale such that significant detail information may be compromised. A second more recent method from [20] trains a joint model to learn depth maps and intrinsic images with activations coupled across iterations. Unlike our approach, the resulting pipeline operates in the gradient domain and requires separate post-processing steps, as well as additional guidance from labeled depth images. Additionally, [24] leverages a generative adversarial network (GAN) to learn intrinsic images from a special resynthesized dataset, but does not provide models or results from existing MPI-Sintel benchmarks for comparison. Finally, [29] trains an encoder-decoder CNN to learn albedo, shading and specular images; however, this approach does not apply to scene-level images as our approach does.

3. Proposed Model

Our proposed model consists of three main parts: an inference net, an edge refinement step, and a domain transform as illustrated in Figure 1. We will now detail each component in turn.

3.1. Inference Net

Unlike previous work that primarily attempts to explicitly predict sparse image gradients [33, 20] or related weight maps [25], the primary parameterized workhorse of our model is an inference net designed to directly predict fil-

tered images in an end-to-end manner. This network contains 20 convolutional neural network (CNN) layers followed by a color correction layer and two gradient computation layers. These are designed as follows:

CNN Layers: Of the 20 CNN layers, the middle 18 are all of the equivalent size $64 \times 64 \times 3 \times 3$, meaning that for each of 64 output feature maps, 3×3 spatial regions of all 64 input feature maps are filtered. In contrast, the first layer takes the input image and represents it as 64 feature maps, while the last layer transforms the multi-channel representation back to the original image space (3 color channels). In both cases the kernel sizes are also 3×3 . Additionally, each convolutional layer is followed by batch normalization (BN) and ReLU layers except for the last one. Finally, to accelerate the training process, we retrofit the middle 16 convolutions with 8 modified residual blocks (see the supplementary file for details of these modifications and network structure).

As observed in [33], a fully convolutional network trained directly in the original image domain may tend to generate very blurry images and unwanted details, while gradient domain supervision is more sensitive to the changes in sharp edges that comply with human perception of contrast more than absolute color values. But we argue that this limitation of the former is largely due to insufficient receptive field size and flexibility. After all, image smoothing and intrinsic image decomposition techniques determine the color values of each pixel in the context of a large surrounding area. Especially for albedo images, small color difference between nearby pixels can accumulate, producing significant differences between distant pixels with complex connecting pathways even when they share the same reflectance. This renders the estimation problem quite difficult if we are restricted to either small or insufficiently parameterized receptive fields. Therefore, instead of using 3 convolutional layers of large kernel size (16×16 , 1×1 and 8×8) as in [33], we adopt 20 convolutional layers with

3×3 kernels, which allows the learned regression function to more accurately account for distant contextual information [30].

Color Correction Layer: Although image smoothing attempts to remove small gradients, it should still generally preserve a high degree of correlation between input and output colors. With so many convolutional layers however, such correlations may eventually weaken without some additional source of long-term memory that reflects the preservation of the original color schema. For this purpose, we include an additional color correction layer to scale each channel independently as follows. Let P_{1_c} denote the output of the 20 layer CNN module associated with color channel c , while I_c is the corresponding input image. We then compute

$$\alpha_c = \arg \min_{\alpha_c} \|I_c - \alpha_c P_{1_c}\|_2 = \frac{P_{1_c} \cdot I_c}{P_{1_c} \cdot P_{1_c}} \quad (2)$$

via a dedicated scaling layer (where \cdot indicates a dot product) and output the updated image prediction $P_{2_c} = \alpha_c P_{1_c}$. As a small caveat, we also threshold values of α_c outside of the range $[0.7, 1.3]$ during training when the predicted images may still be radically different from I_c . Note that network gradients are easily backpropagated through such a layer. Also for testing, we discard any such threshold. A related idea has been incorporated into a classical, optimization-based intrinsic image decomposition [16], but this represents a decidedly different context from our CNN model. In any event, a channel-wise color correction layer is a simple but very effective addition to our pipeline, and we find that it reduces negative color difference effects by a wide margin.

Gradient Supervision: Most image smoothing algorithms seek to more-or-less achieve a piece-wise constant result. Certainly the albedo component of an intrinsic image decomposition problem is well-approximated by such an effect. But smoothing a large region into a single color is a challenging task for convolution since the kernel weights are shared over the whole feature map, but local color values can vary dramatically across regions. To help mitigate this problem, we supervise the x and y gradients $\nabla_x P_2, \nabla_y P_2$ from the color-normalized images P_2 (across all color channels) using a standard auxiliary MSE loss function.

3.2. Edge Refinement and Domain Transform

The images generated by the inference net described above already demonstrate excellent numerical performance. But we observe that some modest color changes still exist in regions that piece-wise constant filters like [5, 32, 34] are able to smooth away. Inspired by [14, 8], we address this lingering issue via a domain transform guided by refined edge estimates. The domain transform is an

edge-preserving processing step that admits efficient implementation via separable 1-D recursive filtering layers applied across rows and columns in an image. Two inputs are required: the raw input signal to be filtered, which corresponds with our predicted image P_2 from the previous layer, and a domain transform density map that differentiates which regions to filter and which to leave unaltered.

This mapping is produced by a small edge refinement subnetwork in our pipeline. The core idea originates from [14], where the density map is simply the original input image gradient. Later [8] experimented with more sophisticated predicted edge maps applied in conjunction with image segmentation scores. However, their pipeline addresses a completely different task, and the embedded, application-specific edge map requires the concatenation of features culled from intermediate CNN layers. In contrast, since we already generate reasonable edge-preserving images from our base inference net, we do not require this degree of sophistication. Hence our proposed edge refinement subnetwork operates as follows. (Further technical details about how to implement domain transforms within a neural network architecture, including gradient backpropagation steps, can be found in [8].)

The edge refinement module consists of two stages. The first computes, for every point in the input P_2 , the averaged absolute differences between each of its four neighboring pixels, which is then further averaged across all color channels to produce a single value. The resulting map is next refined using a very shallow network containing 3 convolutional layers. The feature maps for the first two convolution layers have 64 channels. The kernel size for the 1st and 3rd layer is 15×15 , zero padded with 7 pixels on each side to maintain spatial dimensions, while the kernel size for the 2nd layer is 1×1 , which computes a weighted average of processed pixel vectors for the smoothing operation. The first two convolution layers are followed with ReLU units, while the third is not.

3.3. Modifications for Intrinsic Images

Finally we conclude this section by describing special accommodations for intrinsic image decompositions. Here the inference net is split into two branches after the middle convolutional layer in order to predict albedo and shading images simultaneously. We also observe that (1) can be leveraged as a useful prior for intrinsic image decomposition, enforcing that the estimated decomposition must recombine to approximate the original image I (to the extent that the diffusive, Lambertian model is accurate). Therefore we include an additional term that penalizes deviations of the estimated $A \odot S$ from I , which helps to balance the error between albedo and shading images.

Additionally, intrinsic image decompositions often require potentially larger receptive fields since accumulated

soft shadows sharing the same albedo could cover a wide region. Consequently we extend the depth to 42 convolutional layers and downsample intermediate features maps by 1/2 which saves half the training time. And because albedo and shading images need not have a very close color similarity to the input image, we can remove the color normalization layer for training an intrinsic image decomposition network. For the detailed structure of this modified network, please refer to the supplementary material.

4. Training Details

We begin with the basic image smoothing network depicted in Figure 1. The embedded inference net requires supervision on 4 outputs: the predicted image P_1 directly from the CNN, the color-normalized image P_2 , and computed gradients in x and y directions, i.e., $\nabla_x P_2$, $\nabla_y P_2$. With the $*$ symbol denoting ground truth, the inference net objective terms become

$$l_1(\theta) = w_1 \|P_1 - P_1^*\|_2^2 + w_2 \|P_2 - P_2^*\|_2^2 + w_3 (\|\nabla_x P_2 - \nabla_x P_2^*\|_2^2 + \|\nabla_y P_2 - \nabla_y P_2^*\|_2^2) \quad (3)$$

where θ denotes the parameter set of the network layers and ℓ_2 norm penalties are chosen since they favor high peak signal-to-noise ratios (PSNR), a common evaluation criteria for image estimation problems. After training inference net in isolation, we then learn an independent network for edge refinement, whose output is denoted E with corresponding loss function $\|E - E^*\|_2^2$. Finally, we jointly train the whole network using

$$L(\theta) = l_1(\theta) + w_3 \|E - E^*\|_2^2 + w_4 \|P_3 - P_3^*\|_2^2, \quad (4)$$

where P_3 is the output of the domain transform, and noting that, as a gradient proxy we retain the same weighting parameter w_3 for E as used in Eq. 3 for coordinate-wise gradients.

For the intrinsic image network, the albedo and shading branches share the same loss function as introduced above, referred to as $L_a(\theta)$ and $L_s(\theta)$, except for the removal of supervision on the color-normalized image P_2 and with analogous gradient supervision now denoted to P_1 . We also include supervision that reflects the constraint from Eq. 1, giving the final cost

$$L(\theta) = L_a(\theta) + L_s(\theta) + w_5 \|A_1 \odot S_1 - A^* \odot S^*\|_2^2, \quad (5)$$

where A_1 and S_1 are the estimates obtained from the inference net analogous to P_1 .

We implement all networks within the Torch framework using mini-batch gradient descent, with batch size fixed at 16. The energy function weights from w_1 to w_5 are empirically determined as 0.2, 0.1, 0.35, 0.2, and 0.2/255 respectively. Note that these weights are used for all smoothing

filters and all intrinsic image decompositions, and hence are quite robust; manual tuning in an application-specific manner is not required. Convolutional layers are initialized using the methods from [17]. To train inference net, we use ADAM [21] to accelerate convergence. The learning rate is initially set to 0.01 and then reduced to 0.001 to generate further improvement. For edge refinement, we use simple stochastic gradient descent [23] with learning rate set to 0.01. And for fine-tuning the whole network together, we again use ADAM with a small learning rate of $1e-5$.

5. Experimental Results

This section showcases the numerical and visual advantages of our approach. Further experiments and thorough ablation studies are deferred to the supplementary file.

5.1. Image Smoothing

Dataset: As a low-level vision task, many corpora of natural images are suitable for training and evaluating smoothing algorithms. Here we use the around 17000 natural images from public PASCAL VOC dataset [11] as input, and the corresponding images filtered by traditional smoothing algorithms as the labels/supervision. These images were collected from the flickr photo-sharing website and are in a wide range of viewing conditions, the same data source used by [33] with which we compare. We randomly crop images at a size of 224×224 pixels for a single image to generate about 17,000 patches. To evaluate our performance, we randomly pick 100 images to form our VOC test split. Additional test results using the BSDS500 dataset [1] are similar (see supplementary).

Comparisons: We approximate 8 different image smoothing filters including the bilateral filter (BLF) [27], iterative bilateral filter (IBLF) [13], L_0 smoothing (L_0) [32], rolling guidance filter (RGF) [35], RTV texture smoothing (RTV) [34], weighted least square smoothing (WLS) [12], weighted median filter (WMF) [36] and L_1 smoothing (L_1) [5]. Compared to 7 public models released by [33], we show much better results in Table 1. Likewise, Table 3 presents

	PSNR		SSIM	
	Liu [25]	Ours	Liu [25]	Ours
L_0	32.26	33.97	0.958	0.980
RGF	38.64	38.15	0.986	0.987
WLS	38.29	37.29	0.983	0.986
WMF	33.29	36.68	0.951	0.982
Ave.	35.64	36.52	0.966	0.984

Table 3. Comparison with recent work Liu *et al.* [25] on image smoothing task. Experiments are conducted using their four released trained filters.

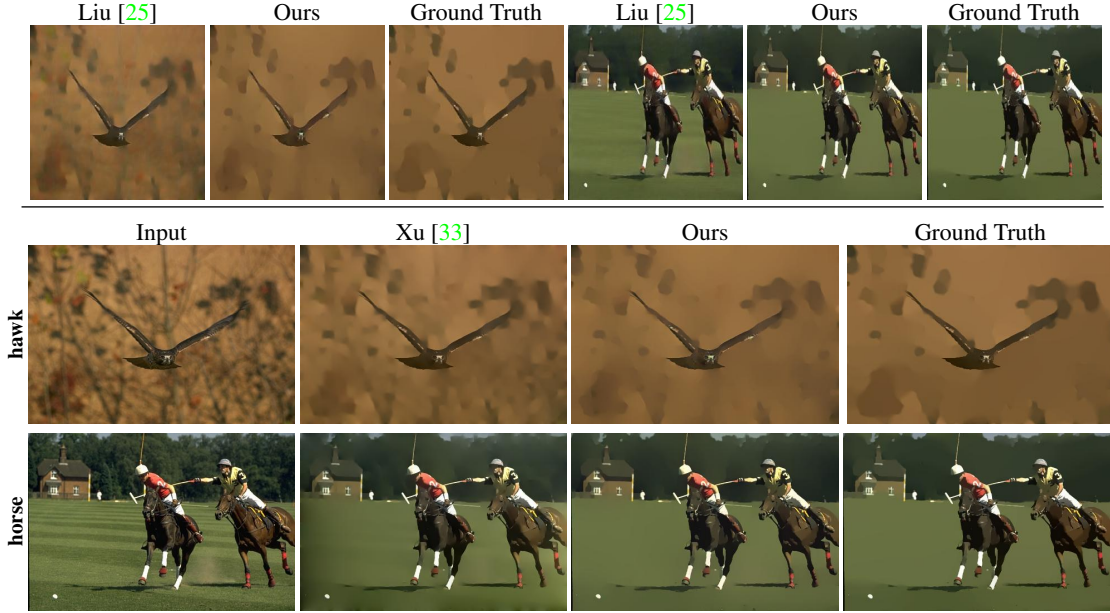


Figure 2. **Qualitative Comparison on Image Smoothing Task.** *Top:* Comparison with Liu *et al.* [25] on the 256×256 image size. *Bottom:* Comparison with Xu *et al.* [33]. All the results are trained on L_0 filter. Zoom in to watch the difference.

		BLF	IBLF	L_0	RGF	RTV	WLS	WMF	L_1	Ave.
PSNR	Xu [33]	35.02	32.97	31.66	32.49	35.68	33.92	29.62		32.62
	Ours	38.34	35.07	33.56	38.62	36.02	38.03	35.75	32.43	36.48
SSIM	Xu [33]	0.976	0.962	0.966	0.950	0.974	0.963	0.960		0.964
	Ours	0.987	0.980	0.981	0.987	0.983	0.985	0.980	0.950	0.983

Table 1. **Quantitative Comparison on Image Smoothing Task.** We report PSNR and SSIM error metrics (larger is better) on 8 different smoothing filters compared with Xu *et al.* [33]. The average value is computed among the front 7 filters. Our results outperform our competitor by a large margin.

comparisons with the recent work from [25]. However, their released code at the time of this writing, which is built upon a multiscale decomposition, cannot run on arbitrary image size, so we test on their default image size 256×256 . Even though we do not apply any special tuning for this size, or train on multiscale images as in Table 3, we still achieve better results on average in their setting. And our consistently higher SSIM values indicate that our learned filters are quite effective in generating perceptually good images. Additionally, we note that WMF can generate relatively blurry images compared to other filters with the parameterization used in these experiments taken from [33], and yet our network handles this case quite well in both tables relative to other approaches. This is likely because previous methods more-or-less directly rely on sparse structures and may have trouble reproducing the softer transitions required by this WMF filter. Visual comparisons with the WMF filter are in the supplementary material.

We next display some visual comparisons in Figure 2 using the L_0 filter, which produces effects more on the piecewise constant end of the smoothing spectrum. Here we ob-

serve that both [33] and [25] have some trouble reproducing constant regions, and [33] has difficulty with erroneous gradient estimates propagating from image boundaries (see horse image). Finally, we evaluate the testing time with traditional edge-aware filters and learned deep networks all on the same computer with public codes and models. Our network achieves favourable speeds and is much faster than most traditional image smoothing methods (see Table 2).

5.2. Intrinsic Image Decomposition

Datasets: We follow two recent state-of-the-art deep learning based methods [31, 20] and evaluate our algorithm on the MPI-Sintel dataset [7] that facilitates scene-level quantitative comparisons. It consists of 890 images from 18 scenes with 50 frames each (except for one that contains 40 images). Due to limited images in this dataset, we randomly crop 10 different patches of size 300×300 from one image to generate 8900 patches. Like [31], we use two-fold cross validation to obtain all 890 test results with two trained models. We evaluate our results on both a *scene split*, where half the scenes are used for training and the other

methods	BLF	IBLF	RGF	L_0	WMF	RTV	WLS	L_1	Xu [33]	Liu [25]	Ours
QVGA (320×240)	0.03	0.11	0.22	0.17	0.62	0.41	0.70	32.18	0.23	0.07	0.06
VGA (640×480)	0.12	0.40	0.73	0.66	2.18	1.80	3.34	212.07	0.76	0.14	0.20
720p (1280×720)	0.34	0.97	1.87	2.43	4.98	5.74	13.26	904.36	2.16	0.33	0.51

Table 2. **Running Time Comparison.** Timing comparisons against different traditional filters and deep learning-based counterparts from Xu *et al.* [33] and Liu *et al.* [25] at various resolutions.

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
<i>image split</i>	Retinex [16]	0.0606	0.0727	0.0667	0.0366	0.0419	0.0393	0.2270	0.2400	0.2335
	Barron <i>et al.</i> [3]	0.0420	0.0436	0.0428	0.0298	0.0264	0.0281	0.2100	0.2060	0.2080
	Chen <i>et al.</i> [9]	0.0307	0.0277	0.0292	0.0185	0.0190	0.0188	0.1960	0.1650	0.1805
	MSCR [31]	0.0100	0.0092	0.0096	0.0083	0.0085	0.0084	0.2014	0.1505	0.1760
	Ours	0.0067	0.0060	0.0063	0.0041	0.0042	0.0041	0.1050	0.0783	0.0916
<i>scene split</i>	MSCR [31]	0.0190	0.0213	0.0201	0.0129	0.0141	0.0135	0.2056	0.1596	0.1826
	Ours	0.0181	0.0175	0.0178	0.0122	0.0118	0.0120	0.1674	0.1382	0.1528

Table 4. **Quantitative Comparison on main MPI-Sintel Benchmark.** We evaluate our results using both scene and image splits across three standard error rates of intrinsic images on the main MPI-Sintel dataset.

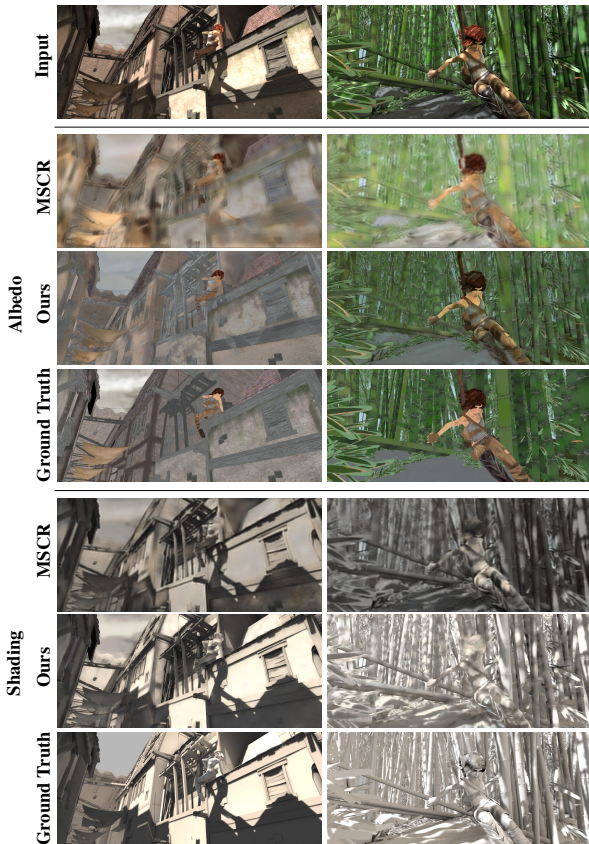


Figure 4. **Qualitative Comparison on main MPI-Sintel Benchmark.** The visual results are evaluated on the model trained on the more difficult *scene split*.

half for testing, and an *image split*, where all 890 images are randomly separated into two parts. Clearly the *scene split*

is less susceptible to inflated results from overfitting, since images in the same sequence has a big overlap with each other while images in different scenes do not.

While investigating the MPI-Sintel dataset online, we noticed that there are actually two sources for the input and albedo images. The first one is obtainable by emailing the authors of [7] directly, which we call main MPI-Sintel dataset given that it appears to be used by most previous methods [9, 31, 24]. But there is also a second, ostensibly lesser-known set that can be partially downloaded from their official web page (but also requires emailing to obtain full ground-truth). We refer to this version as the auxilliary MPI-Sintel dataset, which is used by [20]. Defective images in these data stemming from rendering issues are removed for evaluation purposes following previous methods [31].

Finally, to test performance on real images where scene-level ground-truth is unavailable, we also use the 220 images in the MIT intrinsic dataset [16] as in [31]. This data contains only 20 different objects, each of which has 11 images. To compare with previous methods, we train our model using 10 objects via the split from [3], and evaluate the results on images from the remaining objects.

Comparison: To quantitatively evaluate the performance, we use three criteria, mean-squared error (MSE), local mean-squared error (LMSE), and the dissimilarity version of the structural similarity index (DSSIM) as proposed in [9]. First, in Tables 4 and 5, our model shows much better results on the MPI-Sintel data. Note that the very recent work JCNF [20] benefits from training on additional depth map data, and as a gradient domain method needs to execute a separate post-processing step to reconstruct images.

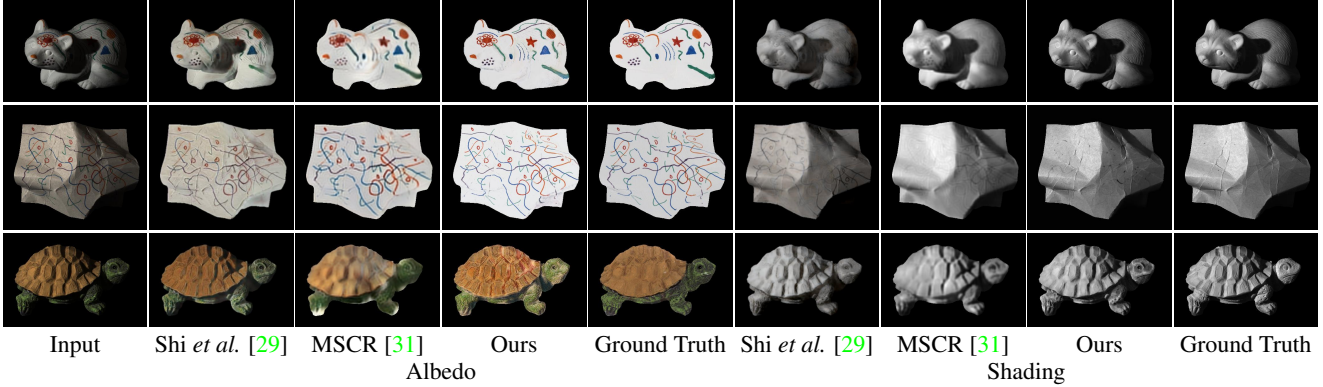


Figure 3. **Qualitative Comparison on MIT Intrinsic Benchmark.** Compared with Shi *et al.* [29] and MSCR [31] on Barron *et al.*’s test split of MIT intrinsic dataset, our algorithm achieves the best/sharpest results.

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
<i>image split</i>	JCNF [20]	0.0070	0.0090	0.0080	0.0060	0.0070	0.0065	0.0920	0.1010	0.0970
	Ours	0.0043	0.0057	0.0050	0.0032	0.0042	0.0037	0.1012	0.0827	0.0919

Table 5. **Quantitative Comparison on auxilliary MPI-Sintel Benchmark.** Similar results to Table 4, except using the auxiliary data. Note that JCNF [20] is only trained and tested on the *image split* of MPI-Sintel dataset, hence our exclusion of the scene split here.

MSCR [31] also benefits from additional training data from the MIT intrinsic dataset.

We show 2 groups of qualitative results trained on the more difficult *scene split* in Figure 4. Note that we are able to predict much sharper and relatively high-quality results even though we have only around 500 training images (and no outside training information as with other methods), and training and testing data do not have any overlap. For more visual results of *image split* or *scene split*, please refer to the supplementary material.

Next, Table 6 presents relative performance using MIT intrinsic data. Here we observe that our approach is significantly better than the other deep networks [29, 31] even though [31] utilizes additional MPI-Sintel data and [29] incorporates additional rendered, large-scale single-object intrinsic images to assist in training. Note that [3] uses a number of specialized priors appropriate for this simplified object-level data, while end-to-end CNN approaches like ours and [31] have less advantage here due to limited training data (110 images). Moreover, [3] is not competitive on other more complex, scene-level data types as shown in Table 4. In Figure 3, our predicted images are also much sharper and more accurate than the other deep learning methods. Finally, we should note that [38] trains a model to predict relative reflectance orderings between image patches and integrates the learned prior within an existing optimization framework for real-world intrinsic image decomposition. We tested their trained model only on the MIT intrinsic data due to the slow running time, and it was not comparable to other methods.

	MSE			LMSE
	albedo	shading	average	total
Zhou <i>et al.</i> [38]	0.0252	0.0229	0.0240	0.0319
Barron <i>et al.</i> [3]	0.0064	0.0098	0.0081	0.0125
Shi <i>et al.</i> [29]	0.0216	0.0135	0.0175	0.0271
MSCR [31]	0.0207	0.0124	0.0165	0.0239
Ours	0.0127	0.0085	0.0106	0.0200

Table 6. **MIT Intrinsic Data Results.** Performance of various methods on Barron *et al.*’s test set [3]. LMSE is computed using an error metric specifically designed for this data [16]. Note also that Barron *et al.*’s approach [3] relies on specialized priors and masked objects particular to this dataset. This approach does not work well on scene level data (see Table 4).

6. Conclusion

This paper proposes a deep network for image smoothing and intrinsic image decomposition tasks. Design features include a deep CNN with large receptive fields operating in the original color domain, a color normalization step, gradient supervision, and a domain transform for refining final image estimates. The resulting pipeline produces edge-preserving smooth images or intrinsic decompositions without any preprocessing, postprocessing, application-specific tuning parameters, or special requirements of input image size. We have also numerically demonstrated state-of-the-art performance on important representative benchmarks.

References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 5
- [2] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013. 1
- [3] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *PAMI*, 37(8):1670–1687, 2015. 7, 8
- [4] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 33(4), 2014. 1
- [5] S. Bi, X. Han, and Y. Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, 34(4):78, 2015. 1, 2, 4, 5
- [6] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics (TOG)*, 33(6):197, 2014. 1
- [7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *ECCV*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 3, 6, 7
- [8] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, 2016. 2, 4
- [9] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013. 1, 7
- [10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015. 3
- [11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, June 2010. 5
- [12] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), Aug. 2008. 1, 5
- [13] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), Aug. 2007. 1, 5
- [14] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 69. ACM, 2011. 2, 4
- [15] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010. 1
- [16] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, pages 2335–2342. IEEE, 2009. 4, 7, 8
- [17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 5
- [18] L. Karacan, E. Erdem, and A. Erdem. Structure-preserving image smoothing via region covariances. *ACM Trans. Graph.*, 32(6):176:1–176:11, 2013. 1
- [19] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 2
- [20] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016. 3, 6, 7, 8
- [21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5
- [22] E. H. Land and J. J. McCann. Lightness and retinex theory. *J. Opt. Soc. Am.*, pages 1–11, 1971. 2
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5
- [24] L. Lettry, K. Vanhoey, and L. Van Gool. Darn: a deep adversarial residual network for intrinsic image decomposition. *arXiv preprint arXiv:1612.07899*, 2016. 3, 7
- [25] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016. 2, 3, 5, 6, 7
- [26] S. H. J. K. M. Aubry, S. Paris and F. Durand. Fast local laplacian filters: Theory and applications. *ACM Transactions on Graphics*, 2014. 1
- [27] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, pages 568–580, 2006. 1, 5
- [28] L. Shen and C. Yeo. Intrinsic images decomposition using a local and global sparse representation of reflectance. In *CVPR*, pages 697–704. IEEE, 2011. 1
- [29] J. Shi, Y. Dong, H. Su, and S. X. Yu. Learning non-lambertian object intrinsics across shapenet categories. *CVPR*, 2017. 3, 8
- [30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4
- [31] M. M. Takuya Narihira and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 3, 6, 7, 8
- [32] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. In *SIGGRAPH Asia*, 2011. 1, 2, 4, 5
- [33] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, pages 1669–1678, 2015. 1, 2, 3, 5, 6, 7
- [34] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via natural variation measure. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2012. 1, 4, 5
- [35] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, pages 815–830, 2014. 1, 5
- [36] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014. 1, 5
- [37] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *PAMI*, 34(7):1437–1444, 2012. 1
- [38] T. Zhou, P. Krahenbuhl, and A. A. Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3469–3477, 2015. 8

Supplementary Material: Revisiting Deep Image Smoothing and Intrinsic Image Decomposition

1 Outline

This document contains several technical details and experiments that could not be included in the main paper because of space considerations.

- **Section 2:** Network structure details, including deep CNN modules and variants for handling intrinsic images.
- **Section 3:** Descriptions of modified residual block used in our deep CNN module.
- **Section 4:** Details of the forward and background propagation process used in the edge refinement network.
- **Section 5:** Self-comparison studies, including both qualitative and quantitative comparison of various network components.
- **Section 6:** Tests of the generalization ability of our 8 learned deep edge-aware filters.
- **Section 7:** More analysis of other deep image smoothing networks.
- **Section 8:** More visual results of different learned edge-aware filters.
- **Section 9:** Additional details regarding intrinsic image decomposition data and error metrics.
- **Section 10:** More visual comparisons with state-of-the-art methods on the intrinsic image decomposition task.

2 Network Structures

In Figure 1 we illustrate the modified network pipeline for handling intrinsic image decompositions, as well as details of the deep CNN modules used by our inference nets (for both image smoothing and intrinsic image decompositions). Decomposing an image into albedo and shading layers requires two outputs from the network, and a much larger receptive field. Our network has been split into two separate branches from the deep CNN module. As shown in the left column of Figure 1, the two split outputs are forwarded through independent edge refinement and domain transform layers to produce albedo and shading images.

Figure 1 also shows the details of the inference net deep CNN modules for both intrinsic image decomposition (middle column) and image smoothing (right column). As observed in [3], distant pixels with the same reflectance on a large surface in intrinsic images could have significant color differences due to the accumulation of smoothly-varying soft cast shadows, which implies that a much larger receptive field may be needed when compared to image smoothing. Therefore, we extend the deep CNN module to 42 convolution layers, and downsample the intermediate feature maps by enlarging the stride of corresponding convolution layers. Note that downsampling can not only increase the receptive field, but also reduce the computational burden caused by extended convolution layers. This is a very effective and efficient training strategy.

For more details of the difference between the two networks, please refer to Sections 3.3 and 4 in the main paper.

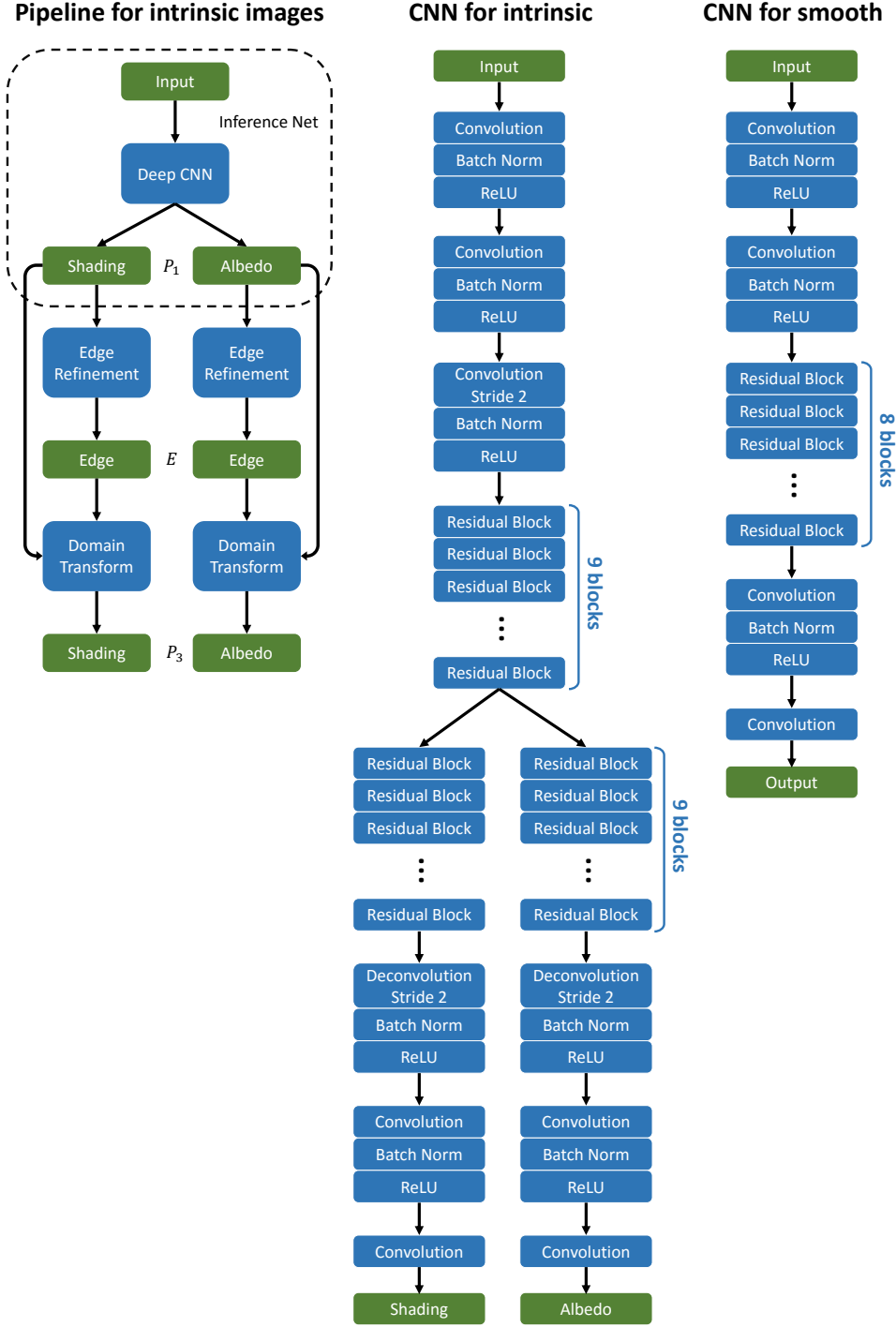


Figure 1: **Network structure details.** *Left column:* Basic pipeline for intrinsic image decompositions (analogous to Figure 1 from the main paper which is for image smoothing). *Middle column:* Deep CNN used by the intrinsic image inference net. *Right column:* Deep CNN used by the image smoothing inference net from Figure 1 in the main paper.

3 Residual Block Details

We apply a slightly modified residual block [6, 7], which we find accelerates the training process and leads to modest improvement in the final performance in most cases. The structure of this block is shown in Figure 2, where the non-linearity represents a variant of ReLU [10] given by

$$f(x) = \begin{cases} 2x & \text{if } x \geq 0, \\ x & \text{otherwise.} \end{cases} \quad (1)$$

Also, BN refers to standard batch normalization. We defer a detailed evaluation of various residual blocks in the context of image smoothing to a future publication.

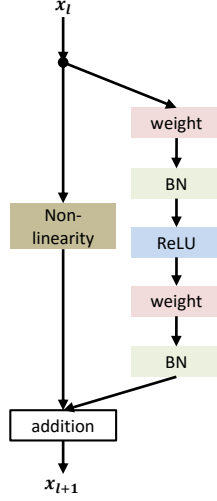


Figure 2: Modified residual block.

4 Forward and Backward Passes for Edge Refinement Network

Here we introduce the forward and backward propagations through the edge computation step at the front end of the edge refinement network. Let us denote the input and output for this module as \mathbf{x} and \mathbf{y} . Then the forward and backward propagations become

$$\mathbf{y}_{i,j} = \frac{1}{2} \sum_c (|\mathbf{x}_{i,j,c} - \mathbf{x}_{i-1,j,c}| + |\mathbf{x}_{i,j,c} - \mathbf{x}_{i+1,j,c}| + |\mathbf{x}_{i,j,c} - \mathbf{x}_{i,j-1,c}| + |\mathbf{x}_{i,j,c} - \mathbf{x}_{i,j+1,c}|) \quad (2)$$

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}_{i,j,c}} = \frac{1}{2} \left[\frac{\partial L}{\partial \mathbf{y}_{i,j}} (\mathbf{h}_{i-1,j,c} + \mathbf{h}_{i+1,j,c} + \mathbf{h}_{i,j-1,c} + \mathbf{h}_{i,j+1,c}) \right. \\ \left. - \left(\frac{\partial L}{\partial \mathbf{y}_{i-1,j}} \mathbf{h}_{i-1,j,c} + \frac{\partial L}{\partial \mathbf{y}_{i+1,j}} \mathbf{h}_{i+1,j,c} + \frac{\partial L}{\partial \mathbf{y}_{i,j-1}} \mathbf{h}_{i,j-1,c} + \frac{\partial L}{\partial \mathbf{y}_{i,j+1}} \mathbf{h}_{i,j+1,c} \right) \right] \quad (3) \end{aligned}$$

$$\mathbf{h}_{i+m,j+n,c} = \begin{cases} 1 & \text{if } \mathbf{x}_{i,j,c} \geq \mathbf{x}_{i+m,j+n,c}, \\ -1 & \text{if } \mathbf{x}_{i,j,c} < \mathbf{x}_{i+m,j+n,c}. \end{cases} \quad \mathbf{m} \in \{1, -1\} \text{ if } \mathbf{n} = 0 \text{ and } \mathbf{n} \in \{1, -1\} \text{ if } \mathbf{m} = 0 \quad (4)$$

respectively, where i and j are image coordinates and c is the color channel index. The gradients will be back propagated to the previous inference network for jointly training the whole pipeline. Initially $\frac{\partial L}{\partial \mathbf{x}_{i,j,c}}$ is set to 0 and $\frac{\partial L}{\partial \mathbf{y}_{i,j}}$ receives value from the subsequent layer.

5 Self-Comparison Study

To better understand how each component of our network contributes to the final results, we evaluate several variants on test data in Table 1. Note that “InferenceNet + DT” is equivalent to our full pipeline, where DT refers to the domain transform. Overall we observe that each component is essential for generating piece-wise constant images, which helps improve PSNR and SSIM by a substantial margin.

Corresponding visual comparisons are shown in Figure 3. The full pipeline composed of all valuable components generates very smooth, sharp and piece-wise constant images. The computed edge map using the output of inference network is processed by the following edge refinement network, and gets much better. Most insignificant details have been eliminated, and salient structures are maintained.

	MSE	PSNR	SSIM
InferenceNet w/o Residual	71.12	30.00	0.961
InferenceNet w/o CC	57.83	31.01	0.973
InferenceNet w/o Gradient	40.38	32.31	0.968
InferenceNet	38.01	32.61	0.973
InferenceNet + DT	30.42	33.56	0.981

Table 1: **Self-Comparison Study.** We justify the effectiveness of each component of our algorithm by removing or adding a single element each time: Residual (residual blocks), CC (color correction), Gradient (gradient supervision), DT (domain transform). The experiments are performed on PASCAL VOC test data and training using the L_0 filter.

6 Generalization Ability of Trained Image Smoothing Networks

To test the generalization ability of our image smoothing networks trained on the PASCAL VOC dataset, we also randomly pick 100 images from the BSDS500 dataset [1] for testing, since as we confirmed from the authors of [13], these data generate comparable results from training with their method on either BSDS500 or self-collected flickr images. However, they also trained slow filters on BSDS500 data which leads to a potential overfitting empirical advantage on our test data.

As shown in Table 2, our results evaluated on BSDS500 test data is still much better than [13]. Our average error is still comparable to that obtained from PASCAL VOC test data, especially for the more important visual perception error metric (SSIM: 0.984 *v.s.* 0.983).

			BLF	IBLF	L_0	RGF	RTV	WLS	WMF	L_1	Ave.
VOC	PSNR	Xu [13]	35.02	32.97	31.66	32.49	35.68	33.92	29.62		32.62
		Ours	38.34	35.07	33.56	38.62	36.02	38.03	35.75	32.43	36.48
	SSIM	Xu [13]	0.976	0.962	0.966	0.950	0.974	0.963	0.960		0.964
		Ours	0.987	0.980	0.981	0.987	0.983	0.985	0.980	0.950	0.983
BSDS	PSNR	Xu [13]	35.31	33.57	31.13	31.88	35.64	35.78	30.35		33.38
		Ours	36.89	35.07	32.98	37.70	36.34	35.89	35.65	32.18	35.79
	SSIM	Xu [13]	0.975	0.966	0.964	0.951	0.975	0.977	0.966		0.967
		Ours	0.986	0.981	0.976	0.986	0.985	0.984	0.983	0.953	0.984

Table 2: **Quantitative Comparison on Image Smoothing Task Using BSDS500 Test Data.** We report PSNR and SSIM error metrics (larger is better) on 8 different smoothing filters compared with Xu *et al.* [13]. The average value is computed among the first 7 filters, for which Xu *et al.* has a model. Our results outperform our competitor by a large margin.

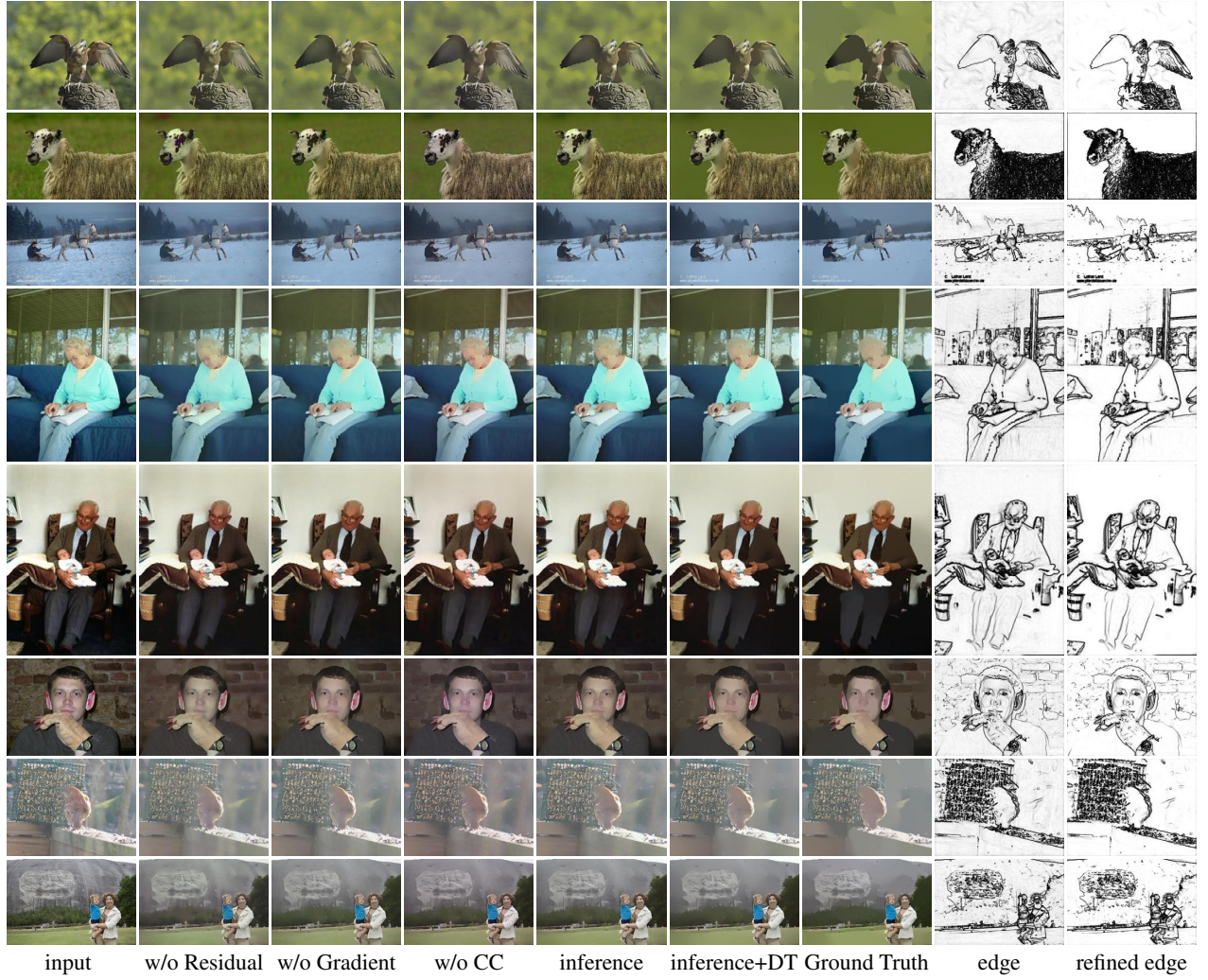


Figure 3: **Qualitative Self-Comparison Study of Our Algorithm.** We test our full pipeline for image smoothing against a series of reduced models with components added or removed in isolation (analogous to Table 1): Residual (residual blocks), Gradient (gradient supervision), CC (color correction), inference (inference net), DT (domain transform), edge (computed edge at the front end of edge refinement network), refined edge (output of edge refinement network). The edge maps are enhanced for better visualization of weak edges.

7 More Analysis Relative to Existing Deep Image Smoothing Networks

This section further analyzes the difference between our algorithm and other deep image smoothing networks through qualitative comparison on L_0 and WMF filters. First, we show some visual results on different learned L_0 filters, which seek piece-wise constant smooth images. The approach from [13] initially predicts gradient maps and then solves an optimized function to reconstruct smooth images from gradient domain to color domain. Since these two steps are separated, their optimization process can be very sensitive to the predicted gradient map. See the “building” and “human” cases in Figure 4, where the gradient error along the image boundary propagates to a large nearby region. Both [13] and [9] use some inflexible filtering or optimization process to reconstruct images, which is not fully extensible to different smoothing algorithms. See the “bird1” and “bird2” cases, where we observe some difficulty in generating piece-wise constant images via these methods.

More visual results are shown in Figure 5, where the goal is to model the WMF filter with parameter settings from [16] that generate somewhat blurry images, instead of piece-wise constant images as L_0 . Existing methods [9, 13], which primarily operate in the gradient domain (or weighted equivalent), may have problems mimicking this type of filtered response. In contrast, our approach directly learns smoothing images with a flexible, parameterized network structure, and is able to handle a wide spectrum of image filters, from those producing relatively blurry effects and some soft transitions, to canonical piece-wise constant outputs.

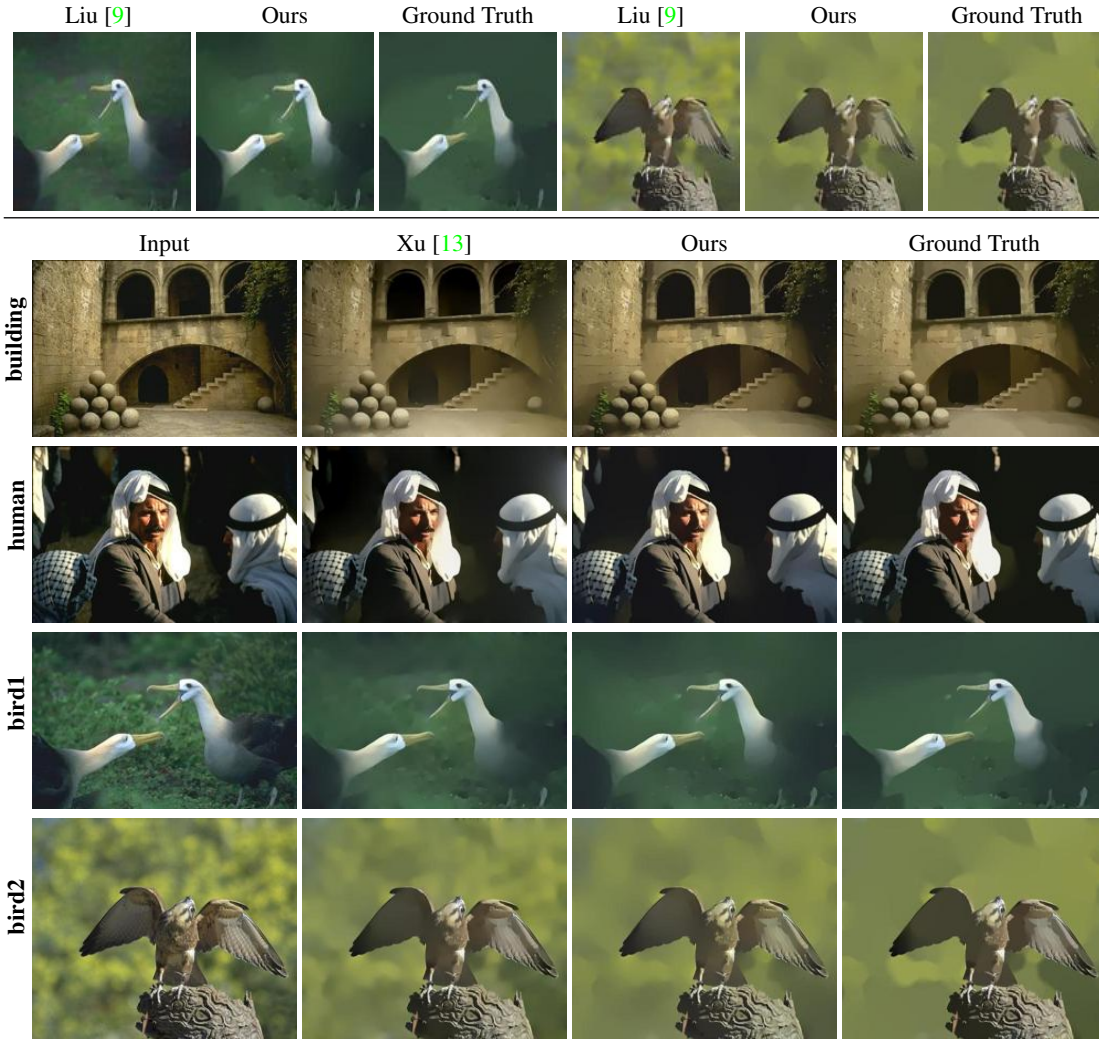


Figure 4: **Qualitative Comparison on the Learned L_0 Filter.** *Top:* Comparison with Liu *et al.* [9] on the 256×256 image size. *Bottom:* Comparison with Xu *et al.* [13]. Zoom in to better see differences.



Figure 5: **Qualitative Comparison on the Learned WMF Filter.** *Top:* Comparison with Liu *et al.* [9] on the 256×256 image size. *Bottom:* Comparison with Xu *et al.* [13]. Zoom in to better see differences.

8 More Results on Image Smoothing

More visual results produced by our trained network mimicking various edge-aware filters are shown in Figures 6-10.

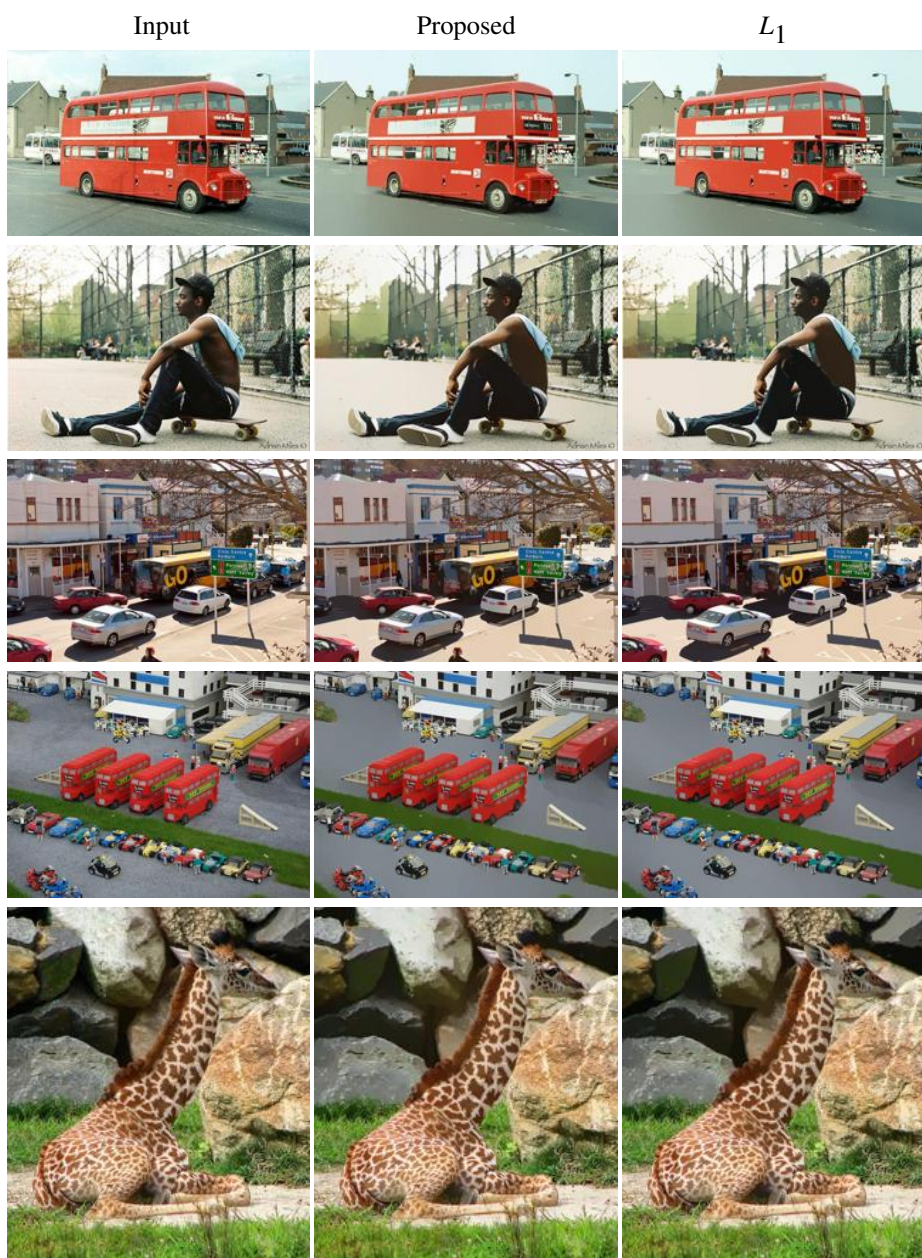


Figure 6: Approximation of L_1 [3] image smoothing algorithm.

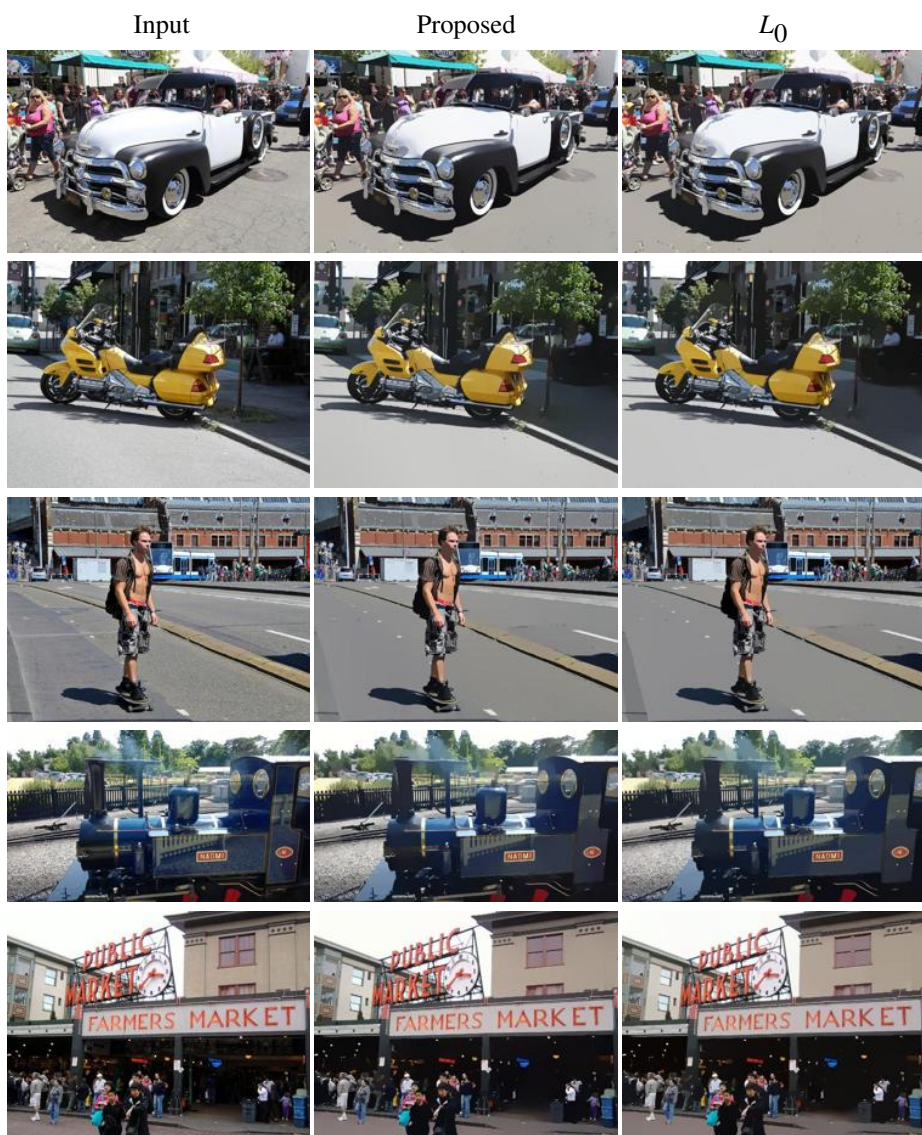


Figure 7: Approximation of L_0 [12] image smoothing algorithm.

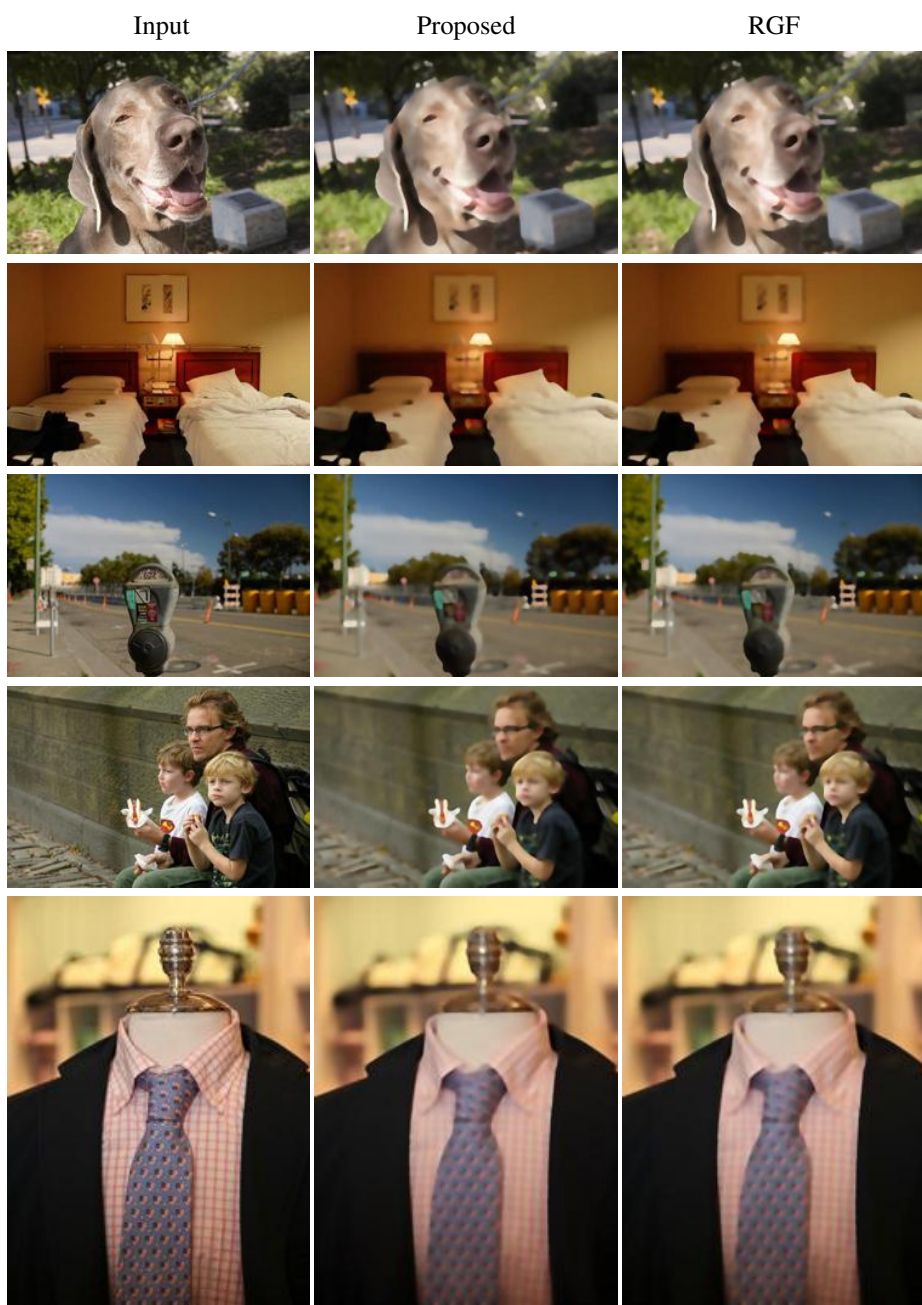


Figure 8: Approximation of RGF [15] image smoothing algorithm.

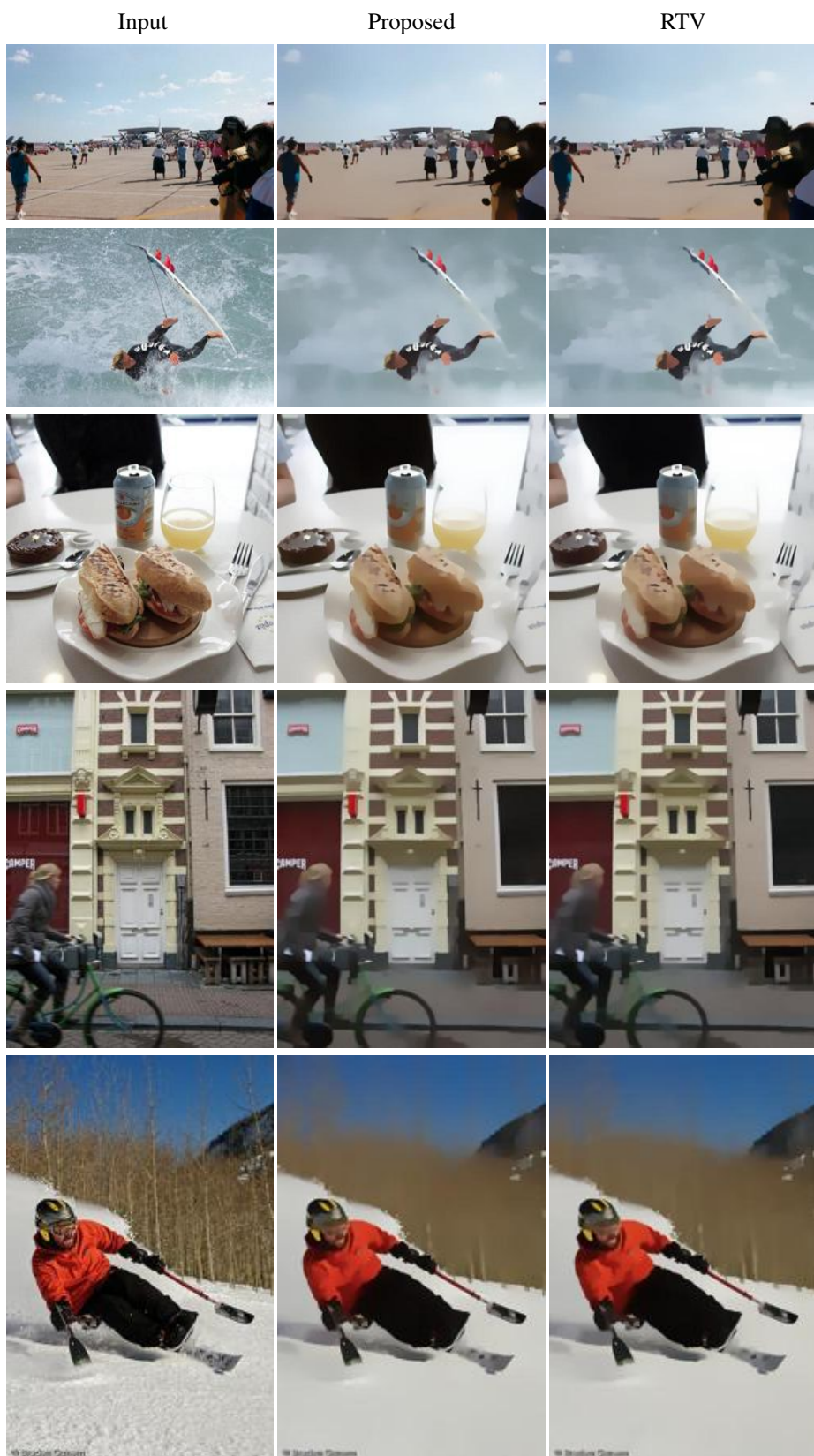


Figure 9: Approximation of RTV [14] image smoothing algorithm.

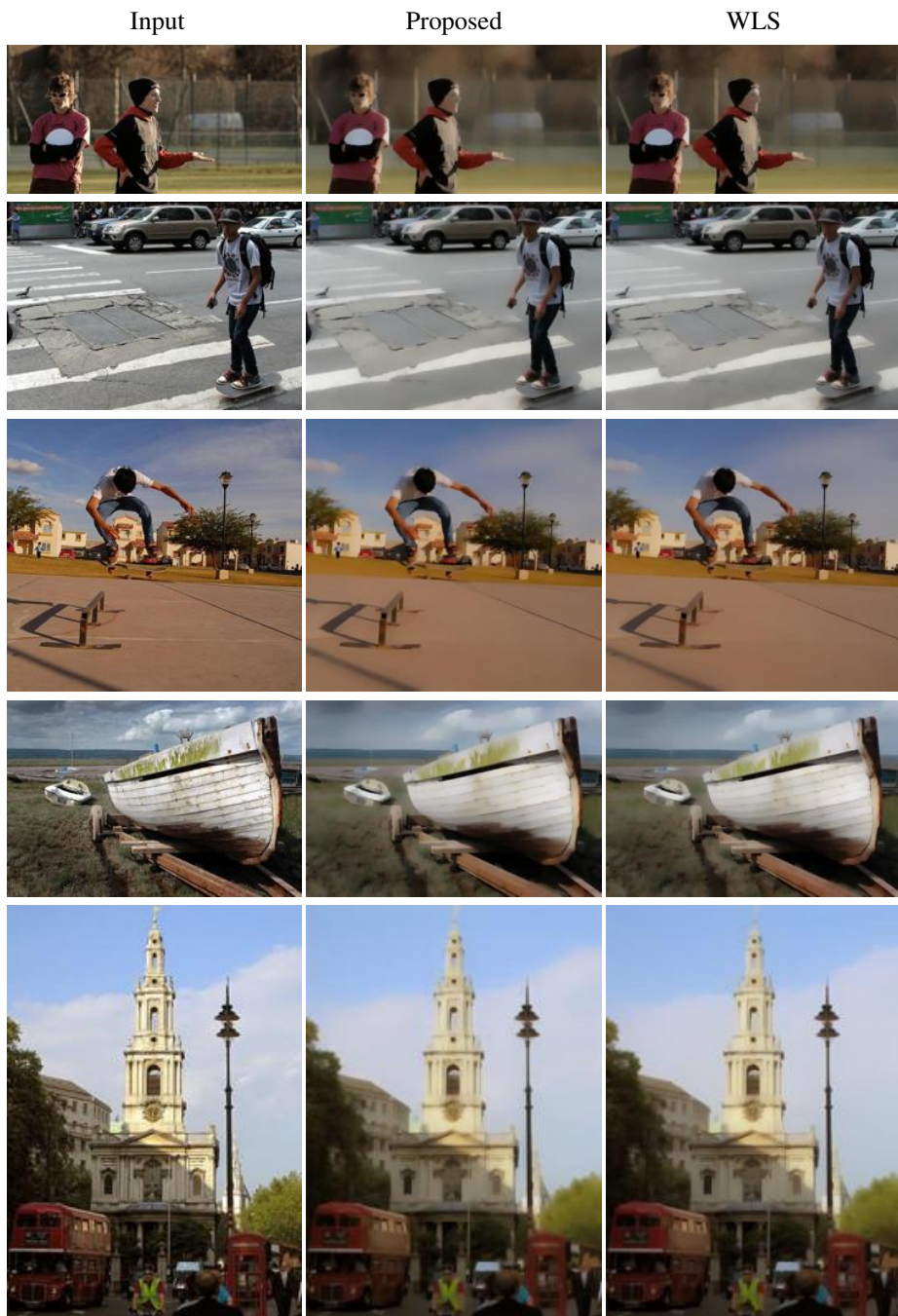


Figure 10: Approximation of WLS [5] image smoothing algorithm.

9 Additional Details Regarding Intrinsic Image Data

We display in Figure 11 visual differences between the *main* and *auxiliary* versions (discussed in the main text) of the MPI-Sintel benchmark. Note also that the main MPI-Sintel benchmark used by most previous methods contains many defective pixels. We remove the defective images for evaluation on the *image split* consistent with [11], and remove two scenes, “bandage_1” and “shaman_3”, for evaluation on the *scene split*. Additionally, to calculate the error on the MIT intrinsic dataset, we use the public evaluation codes in [2] for all methods.

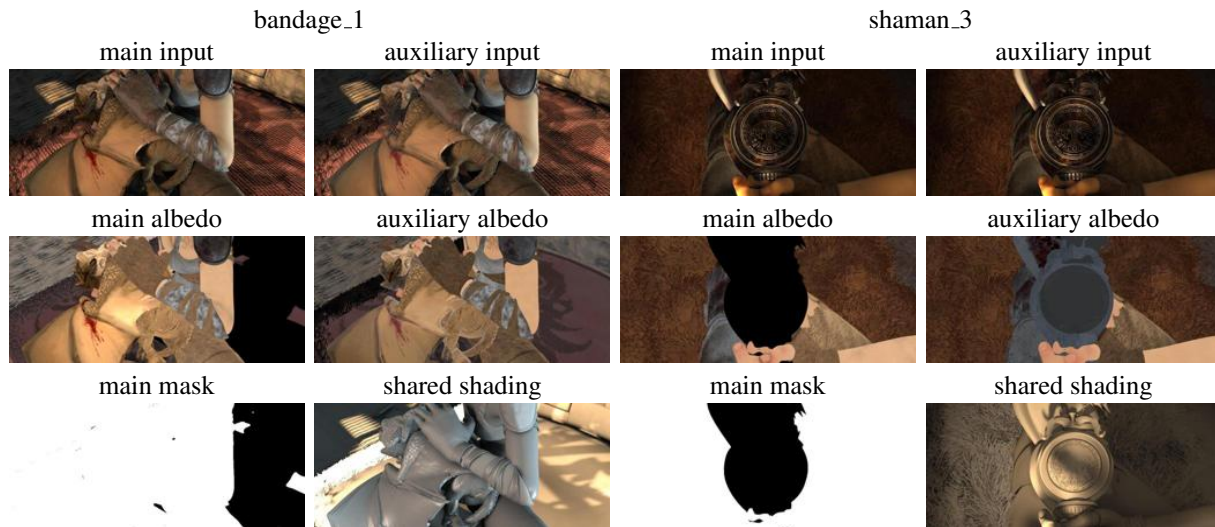


Figure 11: Illustration of differences between the *main* and *auxiliary* versions of the MPI-Sintel benchmark. Note that in the main version, the above albedo images contain many defective pixels due to rendering issues, which is not the case in the auxiliary version. The shading image is shared among both main and auxiliary datasets.

10 More Results on Intrinsic Image Decompositions

This section presents more visual results of our approach compared against existing intrinsic image decomposition methods. More specifically, using the *image split* of the MPI-Sintel benchmark, we provide visual comparisons with many traditional and deep learning based algorithms on the main version of the data in Figures 12, 13, and 14. Additionally, we compare our algorithm with [8] on the auxiliary version of the data in Figure 15, since [8] is the only previous method trained and tested on that version. On the much more difficult *scene split*, we show some qualitative comparisons in Figure 16, 17, and 18.

Our algorithm does not use any third-party training data or information like depth to improve performance. Compared to state-of-the-art methods, our results are much sharper, clearer and more piece-wise constant on both *image split* and *scene split*, main and auxiliary version of MPI-Sintel benchmark.

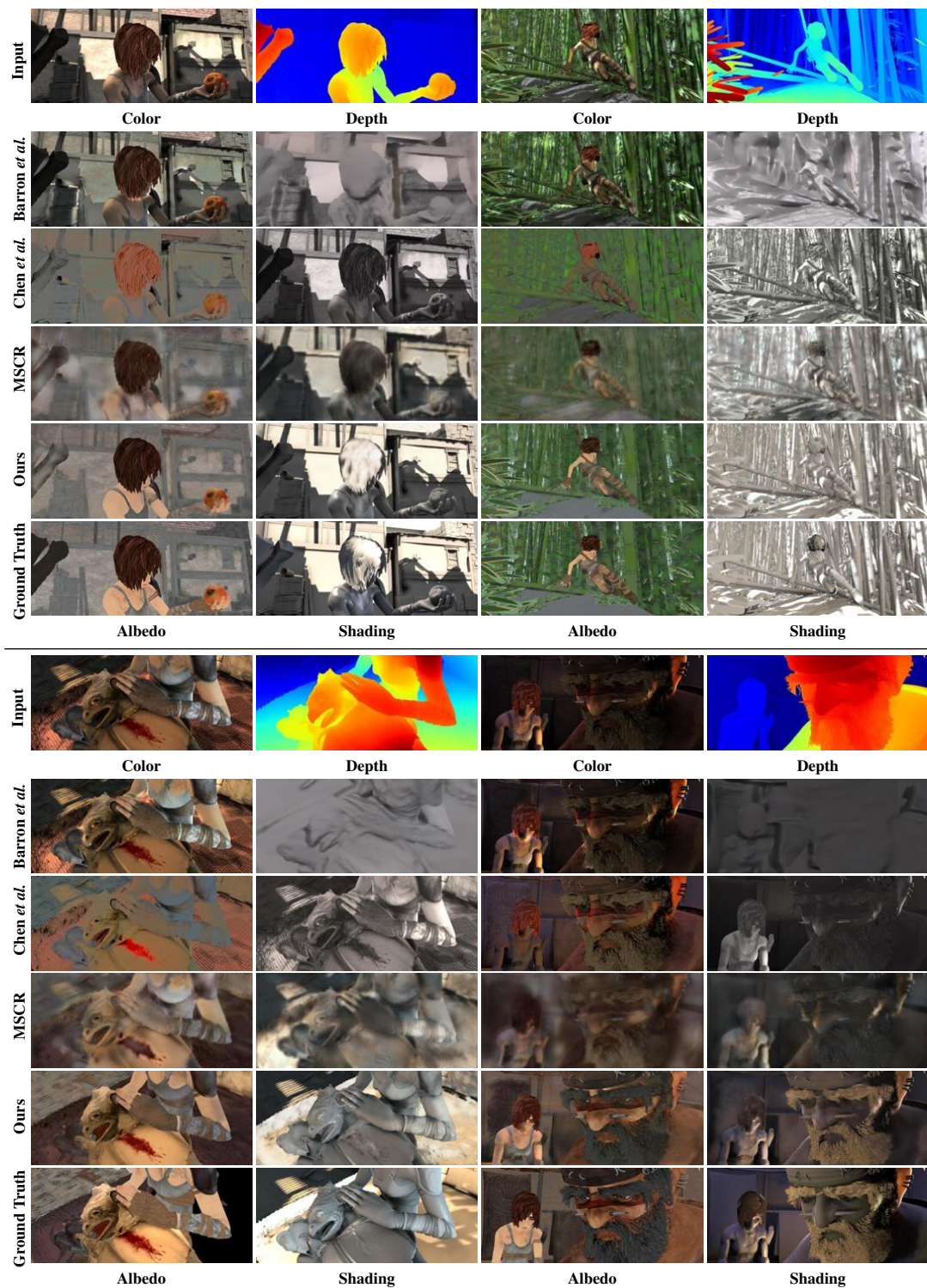


Figure 12: Qualitative comparisons on the *image split* of main MPI-Sintel Benchmark with Barron *et al.* [2], Chen *et al.* [4] and MSCR [11].

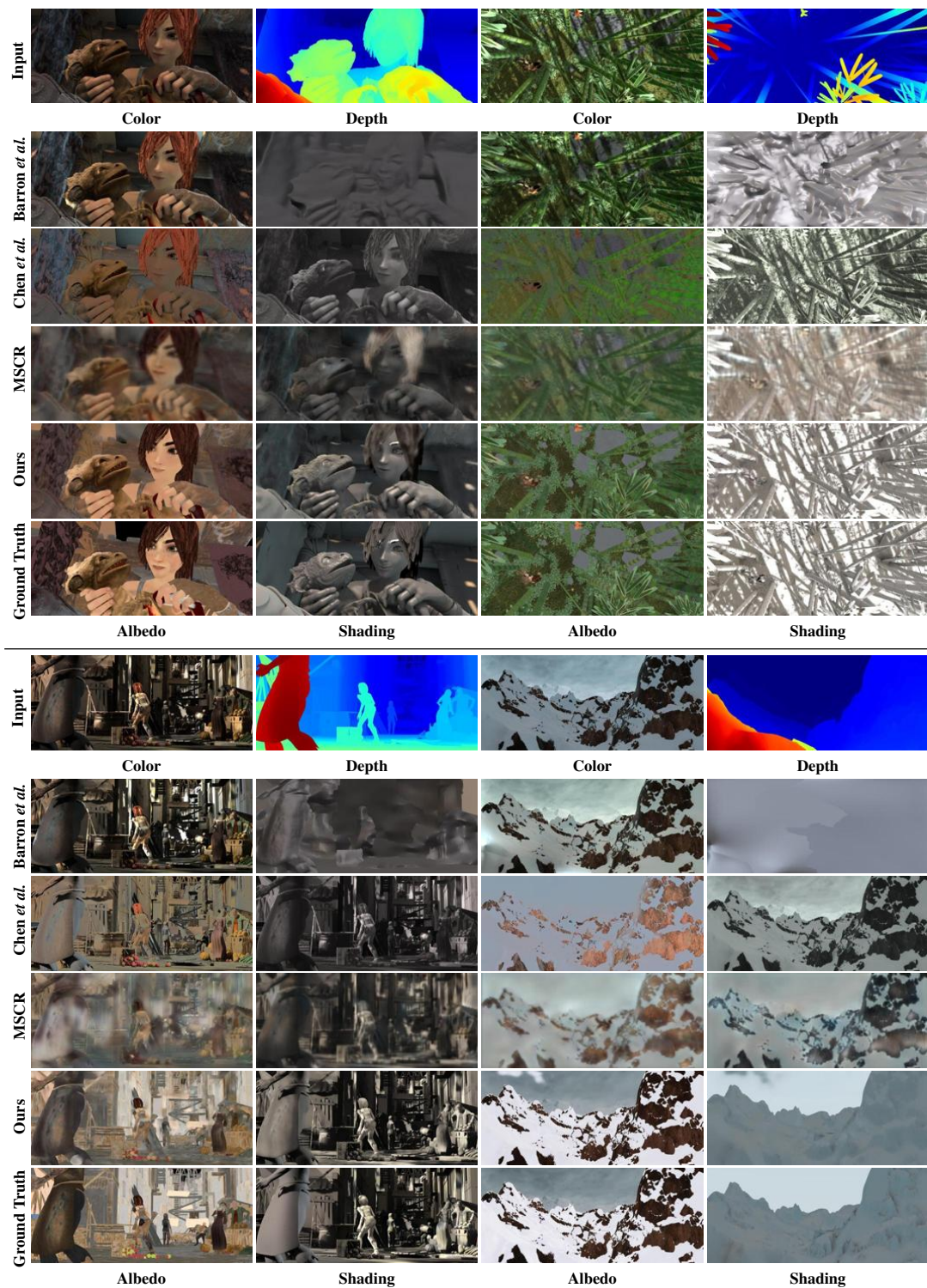


Figure 13: Qualitative comparisons on the *image split* of main MPI-Sintel Benchmark with Barron *et al.* [2], Chen *et al.* [4] and MSCR [11].

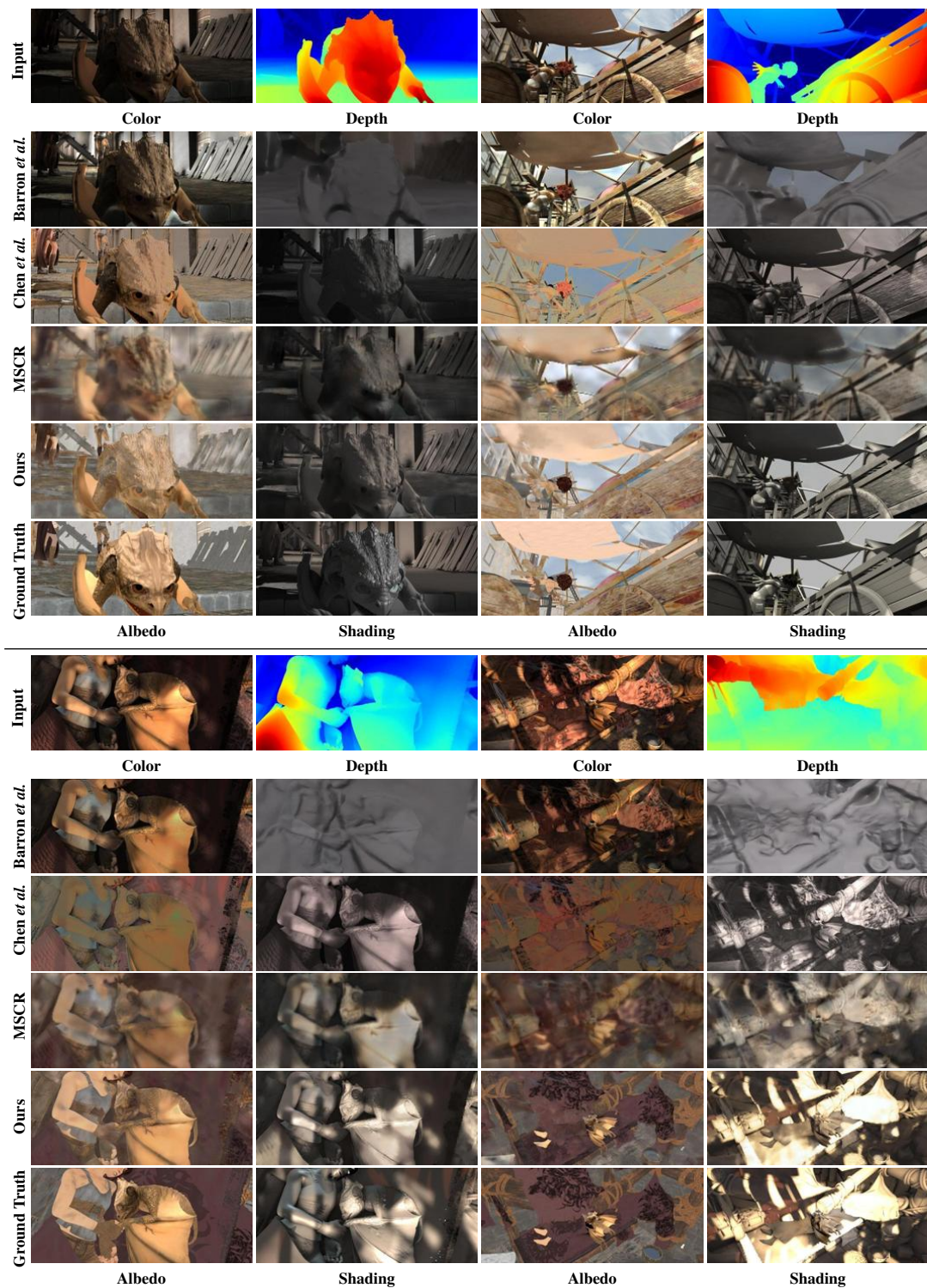


Figure 14: Qualitative comparisons on the *image split* of main MPI-Sintel Benchmark with Barron *et al.* [2], Chen *et al.* [4] and MSCR [11].



Figure 15: Qualitative comparisons on the *image split* of auxiliary MPI-Sintel Benchmark with JCNF [8]

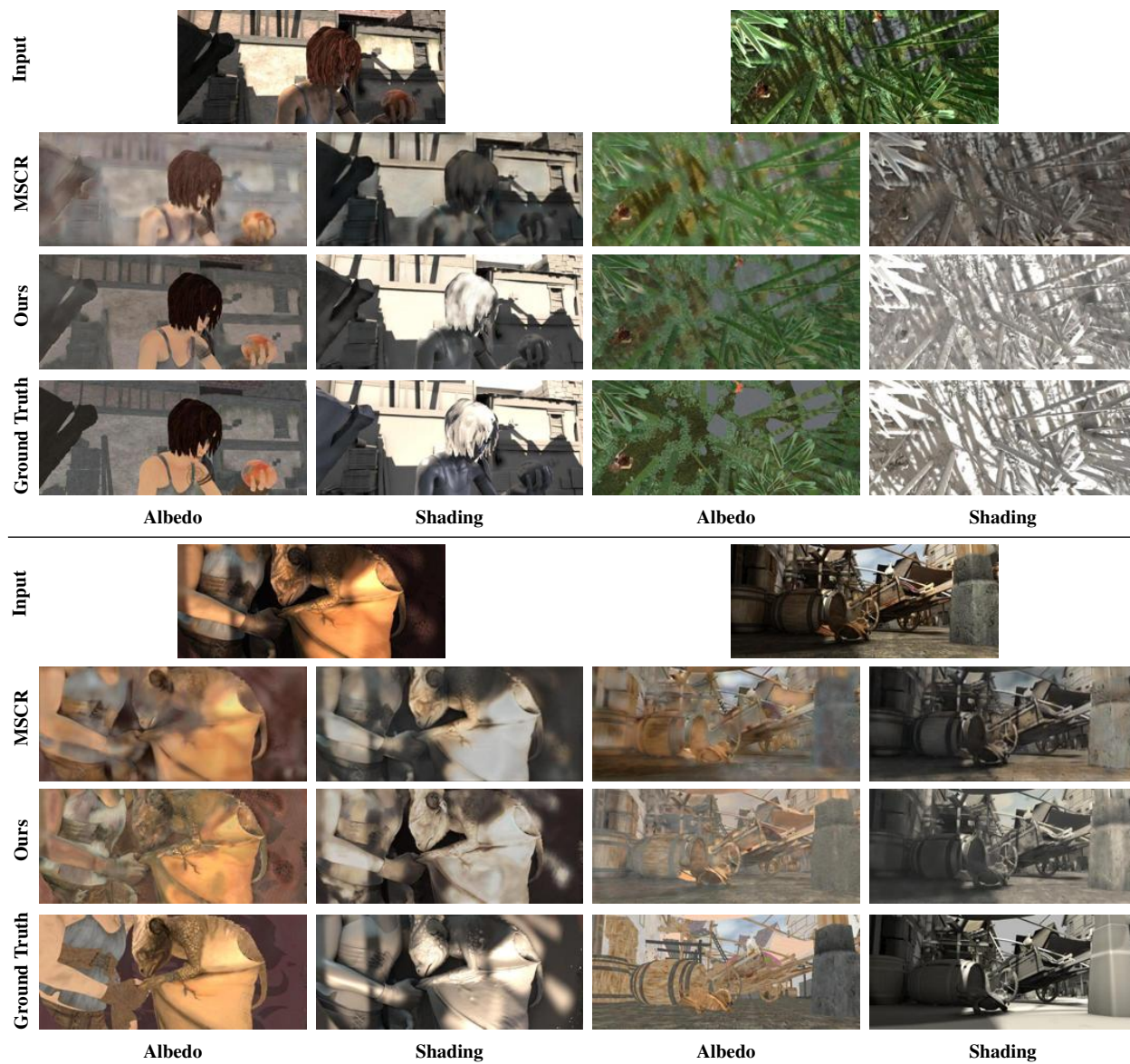


Figure 16: Qualitative comparisons on the more difficult *scene split* of MPI-Sintel Benchmark with MSCR [11].



Figure 17: Qualitative comparisons on the more difficult *scene split* of MPI-Sintel Benchmark with MSCR [11].

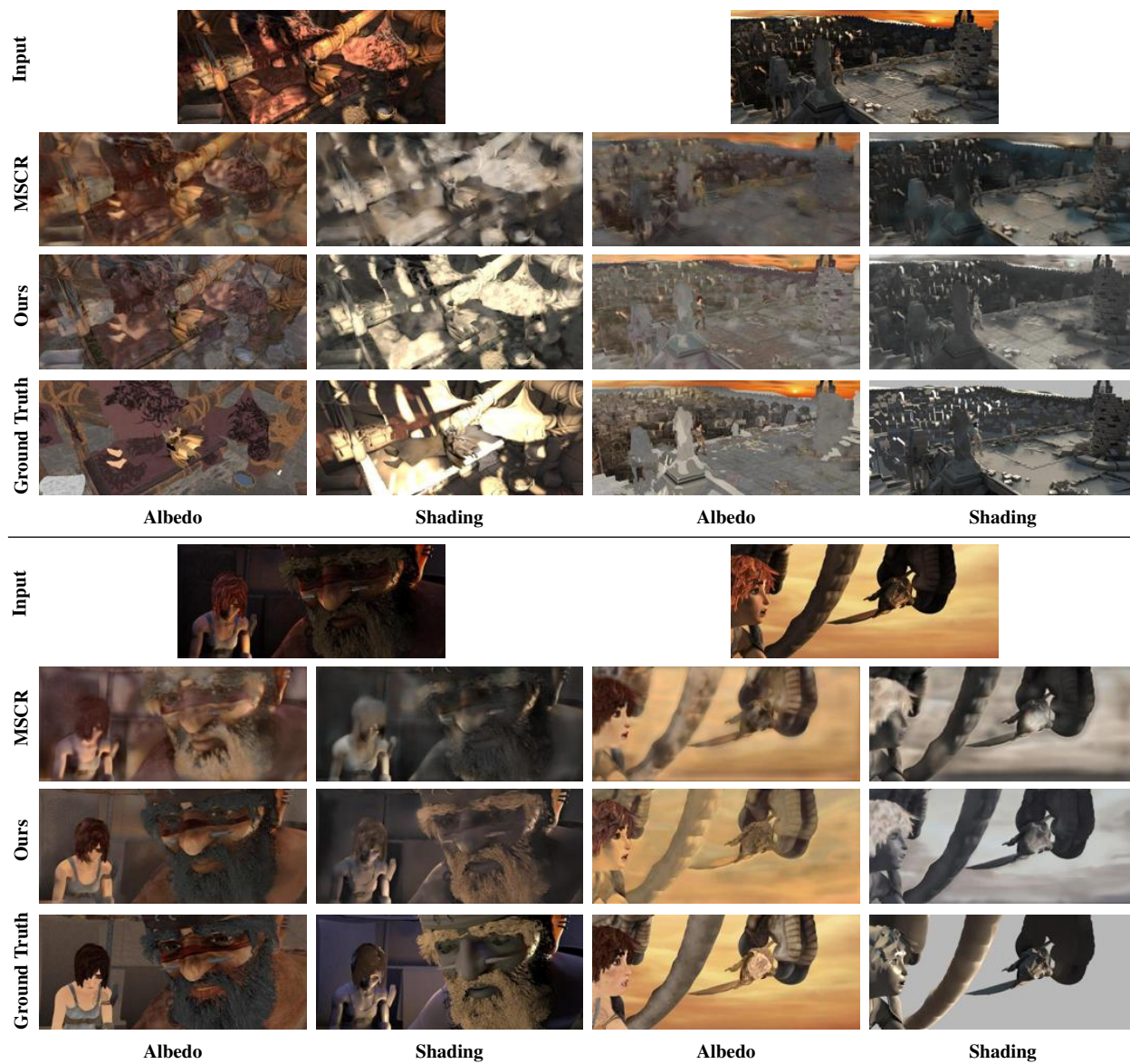


Figure 18: Qualitative comparisons on the more difficult *scene split* of MPI-Sintel Benchmark with MSCR [11].

References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 4
- [2] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *PAMI*, 37(8):1670–1687, 2015. 13, 15, 16, 17
- [3] S. Bi, X. Han, and Y. Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, 34(4):78, 2015. 1, 8
- [4] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013. 15, 16, 17
- [5] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), Aug. 2008. 12
- [6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 3
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *ECCV*, 2016. 3
- [8] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016. 14, 18
- [9] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016. 6, 7
- [10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 3
- [11] M. M. Takuya Narihira and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 13, 15, 16, 17, 19, 20, 21
- [12] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. In *SIGGRAPH Asia*, 2011. 9
- [13] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICCV*, pages 1669–1678, 2015. 4, 6, 7
- [14] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via natural variation measure. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2012. 11
- [15] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, pages 815–830, 2014. 10
- [16] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014. 6